On the Complexity of Simulating Space-Bounded Quantum Computations

John Watrous
Department of Computer Science
University of Calgary
Calgary, Alberta, Canada

February 2, 2004

Abstract

This paper studies the space-complexity of predicting the long-term behavior of a class of stochastic processes based on evolutions and measurements of quantum mechanical systems. These processes generalize a wide range of both quantum and classical space-bounded computations, including unbounded error computations given by machines having algebraic number transition amplitudes or probabilities. It is proved that any space s quantum stochastic process from this class can be simulated probabilistically with unbounded error in space $O(s)$, and therefore deterministically in space $O(s^2)$.

1 Introduction

Let s be a space-constructible function that satisfies $s(n) \in \Omega(\log n)$. Savitch's Theorem [Sav70] gives the following relation between deterministic and nondeterministic space-bounded classes:

$$\text{NSPACE}(s) \subseteq \text{DSPACE}(s^2);$$

a quadratic increase in space compensates for any advantage of nondeterministic computation over deterministic computation. A similar relation holds if nondeterministic computation is replaced with probabilistic computation, even if the probabilistic computation has unbounded error and no restrictions on running time [BCP83, Jun85]. Specifically,

$$\text{PrSPACE}(s) \subseteq \text{DSPACE}(s^2).$$

One expects the same sort of relation to hold when probabilistic computation is replaced by quantum computation, and proving this fact for the most general class of space-bounded quantum computations possible is the purpose of this paper.

A quantum computational analogue of the above relation was proved for a restricted class of quantum computations in [Wat99]. The restrictions on the sorts of quantum computations for which the relation was proved can be informally stated as follows: (i) the measurements permitted in the quantum computations were of a restricted type, and (ii) all transition amplitudes of the machine performing the quantum computation were required to be rational numbers.

The first of these restrictions is a somewhat artificial restriction. Unlike the time-bounded setting in which general measurements can be efficiently simulated by machines allowing only very restricted measurements [AKN98], the effect of such restrictions in the space-bounded case are not well-understood. For instance, it is not even known whether quantum machines can simulate ordinary probabilistic Turing machines in the presence of such restrictions for bounded-error computations. These restrictions are completely removed in this paper—our results hold for the most general type of measurements permitted by the standard model of quantum information.

The second restriction is relaxed to allow any algebraic number amplitudes. The primary motivation for extending the above result to quantum machines with algebraic number amplitudes rather than just rational amplitudes is that algebraic amplitudes arise naturally in many quantum algorithms, so to restrict amplitudes to be rational seems unnatural. For example, one of the simplest and most commonly used quantum operations is the Hadamard transform, which gives transition amplitudes $\pm 1/\sqrt{2}$. While it has been shown that restricting amplitudes to be rational as opposed to algebraic numbers has no effect in the case of polynomial-time bounded-error computations [ADH97], no such result is known to hold for space-bounded quantum computations. The problem one encounters when trying to apply the methodology used in the time-bounded case to the space-bounded case is that accurate approximations of algebraic numbers would need to be obtained using very limited space. This problem of obtaining sufficiently accurate approximations of algebraic numbers in limited space is not solved in this paper; informally speaking, we will use probabilistic approximations of algebraic numbers to circumvent this problem.

A second reason for considering algebraic transition amplitudes is simply that it is an interesting complexity-theoretic question to consider the effect of the choice of amplitudes or probabilities on the computational power of quantum or probabilistic machines. It is well-known that allowing uncomputable numbers as amplitudes or probabilities can allow quantum or probabilistic machines to decide uncomputable languages. What is the effect on computational power for other choices of amplitudes or probabilities? For example, suppose that α is a real, algebraic number contained in the interval $[0,1]$, and suppose L_α is a language that depends in some way on α. It is not unreasonable to imagine that giving a probabilistic Turing machine access to a coin that comes up heads with probability precisely α could allow the language L_α to be decided more efficiently than with just a fair coin, particularly in the unbounded error setting. For the case of algebraic amplitudes/probabilities and unbounded-error space-bounded computations, however, our results imply that no advantage can be gained in this way.

Stated in its most general form, the main result of this paper concerns the complexity of simulating a specific type of stochastic process called a selective quantum process. Quantum computations arising from space-bounded quantum Turing machines and bounded-width quantum circuits (subject to certain uniformity conditions) are special cases of selective quantum processes. In order to demonstrate how the main result may be applied, we will consider a variant of the quantum Turing machine model that allows measurements during its computation. The specific way we choose to define quantum Turing machines in this paper can be viewed as a hybrid of the quantum Turing machine and quantum circuit models; this model will be discussed informally in this section and formalized later in Section 2.3.

Our quantum Turing machines can be viewed as ordinary classical Turing machines augmented with a quantum tape (whose squares contain qubits) along with an internal quantum register of finite size. The classical part of the machine functions as an ordinary Turing machine, but in addition determines the quantum operations performed on the qubits in the internal quantum register and on the quantum tape. The position of the tape head that scans the quantum tape will be classical, so the effect of the computation on the quantum tape is similar to the action of a quantum circuit on a collection of qubits. The quantum operations may produce measurement results that are visible to the classical part of the machine and may influence its future actions. We assume the classical finite state control includes an accepting and a rejecting state that determines the outcome of the computation.

The quantum operations performed by a quantum Turing machine can be described by collections of matrices in a standard way known as the operator-sum representation. We will discuss this representation in detail in Section 2.2. Our Turing machines will be restricted to performing only those quantum transformations that can be described in this way by matrices whose entries are algebraic numbers. (This is what is meant by saying that the machine has algebraic transition amplitudes.)

For any space-bound s we will define a complexity class $\text{PrQSPACE}_{\mathbb{A}}(s)$ as follows. A language L is in $\text{PrQSPACE}_{\mathbb{A}}(s)$ if and only if there exists a quantum Turing machine as described above that, on each input x, visits at most $O(s(|x|))$ tape squares on its work tape and on its qubit tape, and operates as follows: if $x \in L$, then the machine accepts with probability strictly larger than $1/2$, while if $x \notin L$ the machine accepts with probability at most $1/2$. (Similar to the definition of quantum Turing machines, a more formal definition of this class appears in Section 2.3.) The following theorem is obtained.

Theorem 1. *For any space-constructible function $s(n) \in \Omega(\log n)$, we have*

$$\text{PrQSPACE}_{\mathbb{A}}(s) = \text{PrSPACE}(s).$$

Corollary 2. *For any space-constructible function $s(n) \in \Omega(\log n)$, we have*

$$\text{PrQSPACE}_{\mathbb{A}}(s) \subseteq \text{DSPACE}(s^2).$$

The main techniques used in this paper have been developed in previous papers on classical space-bounded computation [AO96, BCP83, Jun85]. In particular we rely on the theory of GapL functions, which were first defined by [AO96] as a logarithmic space variant of the GapP functions of [FFK94]. GapL functions will allow us to indirectly perform computations on matrices that govern the quantum computations or quantum processes being considered in a space-efficient manner. GapP functions were used in this way in the quantum setting by [FR99]. Properties of GapL functions are also exploited to handle algebraic transition amplitudes. It is shown that any real algebraic number can be efficiently approximated by a ratio of GapL functions, which allows the algebraic amplitudes to be incorporated into the general method.

The remainder of this paper is organized as follows. In Section 2 we discuss relevant facts from classical space-bounded complexity and quantum information. This section includes a definition of selective quantum processes and a more formal discussion of the quantum Turing machine model described above. Section 3 is devoted to proving that real algebraic numbers can be approximated by ratios of GapL functions. In Section 4 this fact is used to prove a technical lemma that abstracts the problem of simulating space-bounded quantum processes to a problem concerning infinite series of matrix powers. This lemma is then applied in Section 5 to the problem of simulating selective quantum processes. Finally, in Section 6 these results are applied to the quantum Turing machine model we define in order to obtain Theorem 1. We conclude with Section 7, which briefly mentions some possible directions for further research on space-bounded quantum computation.

2 Definitions and Background Information

2.1 Space-bounded complexity and GapL functions

In this section we review some facts concerning space-bounded complexity. It is assumed the reader is already familiar with deterministic, nondeterministic, and probabilistic Turing machines, and with the basics of space-bounded computation. However, we will take a moment to review particular definitions for deterministic, nondeterministic, and probabilistic Turing machines, which will serve to make our assumptions more precise and to help illustrate the relation of our definition of quantum Turing machines to classical Turing machines. For background information on space-bounded computation, we suggest the survey paper of [Sak96].

We will be primarily concerned with Turing machines that test membership in languages, as opposed to Turing machines that compute functions. For the moment let us restrict our attention to Turing machines that test language membership. Such Turing machines will have two tapes: a read only input tape and a read/write work tape, both two-way infinite. (When we discuss quantum Turing machines later in Section 2.3, there will be an additional quantum tape.) Let Σ denote the input alphabet and Γ the work tape alphabet for a given Turing machine, and let Q denote the set of internal states. The blank symbol # is assumed to not be included in the input and work tape alphabets. The initial state is q_{init}, and there are two special halting states $q_{acc}, q_{rej} \in Q \backslash \{q_{init}\}$ that indicate acceptance and rejection of the computation, respectively.

The transition function of a deterministic Turing machine has the form

$$\delta : (Q \backslash \{q_{acc}, q_{rej}\}) \times (\Sigma \cup \{\#\}) \times (\Gamma \cup \{\#\}) \rightarrow Q \times (\Gamma \cup \{\#\}) \times \{-1, 0, +1\}^2$$

while the transition function of a nondeterministic or probabilistic Turing machine has the form

$$\delta : (Q \backslash \{q_{acc}, q_{rej}\}) \times (\Sigma \cup \{\#\}) \times (\Gamma \cup \{\#\}) \rightarrow \mathcal{P}\left(Q \times (\Gamma \cup \{\#\}) \times \{-1, 0, +1\}^2\right)$$

(where $\mathcal{P}(\cdot)$ denotes the power set). For the nondeterministic and probabilistic cases, it is required that $|\delta(s, \sigma, \tau)| \in \{1, 2\}$. The interpretations of such transition functions is the usual one. In the nondeterministic case the restriction $|\delta(s, \sigma, \tau)| \in \{1, 2\}$ means that either one or two nondeterministic choices is available at each step, and in the probabilistic case it means that transition probabilities are restricted to be in the set $\{1/2, 1\}$.

A Turing machine M runs in space s if the following conditions hold for every computation path of the computation on every input string $x \in \Sigma^*$.

1. The input tape head never leaves the region contained inclusively between the tape square immediately preceding the first input symbol and the tape square immediately following the last input symbol.

2. The work tape head never moves to the left of its initial position and moves fewer than $s(|x|)$ squares to the right of its initial location. (This implies that the total number of work tape squares visited is at most $s(|x|)$.)

It should be mentioned that in the probabilistic case this definition of space-usage is problematic when studying non-space-constructible space bounds; see [Gil77] for further details. We will only be concerned with space bounds s that are space-constructible, and moreover satisfy $s(n) \in \Omega(\log n)$.

Definition 3. A language L is in DSPACE(s) if and only if there exists a deterministic Turing machine M running in space $O(s)$ that accepts every input string $x \in L$ and does not accept any input string $x \notin L$.

A language L is in NSPACE(s) if and only if there exists a nondeterministic Turing machine M running in space $O(s)$ that satisfies the following. For every input string $x \in L$ there exists a computation path of M on input x leading to acceptance, and for every input string $x \notin L$ there are no computation paths of M on x leading to acceptance.

A language L is in $\mathrm{PrSPACE}(s)$ if and only if there exists a probabilistic Turing machine M running in space $O(s)$ that satisfies the following. For every input string $x \in L$ the probability that M accepts x is greater than $1/2$, and for every input string $x \notin L$ the probability that M accepts x is at most $1/2$.

The following abbreviations are used for classes based on logarithmic space-bounds:

$$\begin{aligned} \mathrm{L} &= \mathrm{DSPACE}(\log), \\ \mathrm{NL} &= \mathrm{NSPACE}(\log), \\ \mathrm{PL} &= \mathrm{PrSPACE}(\log). \end{aligned}$$

In the previous definition no restrictions have been placed on whether or not M halts along all computation paths; rejection and running forever are equivalent for the purposes of this definition. However, these particular classes are robust in this sense, meaning that one can impose the additional constraint that M has no infinite computation paths on any input without changing the classes. This is not necessarily the case for all space-bounded classes—see [Sak96] for a further discussion of this issue.

As mentioned above, one may also consider Turing machines that compute functions as opposed to testing language membership. In this case, the machine is equipped with an additional output tape that is one-way infinite, write-only, and whose tape head may not move left. It will only be necessary for us to consider a very restricted type of such machines: they will be deterministic, have output alphabet $\{0,1\}$, and run in logarithmic space. The output is interpreted as a nonnegative integer written in binary, and the output tape does not contribute to the space-usage of the machine. A function $f : \Sigma^* \rightarrow \mathbb{N} = \{0,1,2,\ldots\}$ is said to be in FL if there exists a deterministic Turing machine that halts on all inputs and outputs $f(x)$ on input x for all $x \in \Sigma^*$.

Next we briefly review the notion of counting complexity, which has its origins in the work of [Val79] and has had several applications in complexity theory, including in quantum computing [FGHP99, FR99]. For further information on counting complexity, see the survey of [For97]. Counting complexity was applied to space-bounded computation in [AO96].

Consider a nondeterministic Turing machine M running in logarithmic space. On each input x there is some number of computation paths that lead to accepting configurations and some number of paths that lead to rejecting configurations. Denote these numbers by $\#M(x)$ and $\#\overline{M}(x)$, respectively. For the purposes of the following definition we restrict our attention to nondeterministic machines having a finite number of computation paths on every input. (Equivalently we may say that $\#M(x)$ is undefined if M has infinitely accepting paths on input x, and similar for $\#\overline{M}(x)$ and the number of rejecting paths.)

Definition 4. A function $f : \Sigma^* \rightarrow \mathbb{Z}$ is a GapL function ($f \in \mathrm{GapL}$) if and only if there exists a logarithmic space nondeterministic Turing machine M such that $f(x) = \#M(x) - \#\overline{M}(x)$ for every input x.

The following theorem was proved by [AO96].

Theorem 5. *Let $L \subseteq \Sigma^*$. Then $L \in \mathrm{PL}$ if and only if there exists $f \in \mathrm{GapL}$ such that $x \in L \Leftrightarrow f(x) > 0$ for every $x \in \Sigma^*$.*

Other space-bounded complexity classes can be characterized in terms of GapL functions in a similar way, but we will only need the GapL characterization of PL in this paper.

One of the attractive features of the class of GapL functions is that it possesses strong closure properties, which can be very helpful for showing that certain problems are contained in PL (or other complexity classes characterized by GapL functions).

Theorem 6. *Let $f \in \mathrm{GapL}$, let $g \in \mathrm{FL}$, and let p be an integer polynomial. Define functions a, b, and c as follows:*

$$a(x) = f(g(x)), \quad b(x) = \sum_{i=0}^{p(|x|)} f(x,i), \quad \textit{and} \quad c(x) = \prod_{i=0}^{p(|x|)} f(x,i).$$

Then a, b, and c are in GapL.

Note that any integer function computable in logarithmic space is therefore necessarily a GapL function: $\mathrm{FL} \subseteq \mathrm{GapL}$.

Remark 7. In Theorem 6, and elsewhere in this paper, two standard conventions are being applied: (i) the number i is encoded as a string using binary notation, and (ii) $f(x,i)$ is shorthand for $f(\langle x,i\rangle)$, where $\langle\cdot,\cdot\rangle$ is any standard pairing function suitable for discussing logarithmic space bounded computations. A similar convention is applied whenever a GapL function takes more than two input strings.

Finally, the following theorem due to [AAM03], which establishes a close relationship between GapL functions and the determinant function, will be a key fact for our main result.

Theorem 8. *Let $f \in \mathrm{GapL}$ and let p be a positive integer polynomial.*[1] *For each $x \in \Sigma^*$ let $A(x)$ be a $p(|x|) \times p(|x|)$ matrix defined as $A(x)[i,j] = f(x,i,j)$, and define $g(x) = \det(A(x))$. Then $g \in \mathrm{GapL}$.*

2.2 Selective quantum processes

In this section we state the definition of selective quantum processes that will be used throughout the paper. For background on quantum information see, for instance, [KSV02] and [NC00]. Our definition of selective quantum operations is implicit in [BDE$^+$98] and [NC97].

Suppose that we have a finite set S that we associate with the classical states of a given system. For example, in the case of quantum circuits the set S may be the set of all 0-1 assignments to the wires at some particular level in the circuit, while in the case of quantum Turing machines S may be the set of all configurations of the machine subject to some given space-bound. Quantum states of such a system are represented by density matrices, which are positive semidefinite matrices in $\mathbb{C}^{S\times S}$ having unit trace. We will denote the set of all such density matrices by $\mathbf{D}(\mathbb{C}^{S\times S})$.

A *selective quantum operation* is a mapping that takes as input a density matrix $\rho \in \mathbf{D}(\mathbb{C}^{S\times S})$ and outputs a probability distribution over pairs of the form (τ, ρ_τ), where τ is a symbol from some alphabet Δ and $\rho_\tau \in \mathbf{D}(\mathbb{C}^{S\times S})$. We refer to τ as the classical output of the operation; this may be the result of some measurement performed on ρ, but this is not the most general situation. In order to be physically realizable (in an idealized sense), a selective quantum operation $\mathcal{E}$ must be described by a collection

$$\{A_{\tau,k} \,:\, \tau \in \Delta,\, 1 \leq k \leq m\} \subseteq \mathbb{C}^{S\times S}$$

(where m is an arbitrary positive integer) that satisfies the constraint

$$\sum_{\tau\in\Delta} \sum_{k=1}^{m} A_{\tau,k}^* A_{\tau,k} = I. \tag{1}$$

[1]By a *positive integer polynomial* we mean an integer polynomial p such that $p(n)$ is a positive integer for every nonnegative integer n.

To such a collection of matrices we associate a function $p_\tau : \mathbf{D}(\mathbb{C}^{S\times S}) \rightarrow [0,1]$ and a (partial) function $\Phi_\tau : \mathbf{D}(\mathbb{C}^{S\times S}) \rightarrow \mathbf{D}(\mathbb{C}^{S\times S})$ as follows:

$$\begin{aligned} p_\tau(\rho) &= \operatorname{Tr}\left(\sum_{k=1}^{m} A_{\tau,k}\rho A_{\tau,k}^*\right) \\ \Phi_\tau(\rho) &= \frac{1}{p_\tau(\rho)}\sum_{k=1}^{m} A_{\tau,k}\rho A_{\tau,k}^*. \end{aligned}$$

(In case $p_\tau(\rho) = 0$, $\Phi_\tau(\rho)$ is undefined.) Now, on input ρ, the output of the selective quantum operation $\mathcal{E}$ described by $\{A_{\tau,k} : \tau \in \Delta,\ 1 \leq k \leq m\}$ is defined to be $(\tau, \Phi_\tau(\rho))$ with probability $p_\tau(\rho)$ for each $\tau \in \Delta$. The constraint 1 implies that $0 \leq p_\tau(\rho)$ and $\sum_\tau p_\tau(\rho) = 1$ for any density matrix ρ, and furthermore that each $\Phi_\tau(\rho)$ is a density matrix whenever $p_\tau(\rho) > 0$.

It will be helpful to also associate with $\{A_{\tau,k}\}$ a function Ψ_τ for each $\tau \in \Delta$ as follows:

$$\Psi_\tau(\rho) = \sum_{k=1}^{m} A_{\tau,k}\rho A_{\tau,k}^*.$$

Each $\Psi_\tau(\rho)$ is just an unnormalized version of $\Phi_\tau(\rho)$, i.e., a positive semidefinite matrix whose trace may be less than 1. It will simplify matters when calculating unconditional probabilities to consider these functions.

Finally, a *selective quantum process* is a stochastic process $\{R_1, R_2, \ldots\}$, where each R_t is a random object (taking values in Δ) whose values correspond to the classical outputs of a selective quantum operation. Specifically, a selective quantum operation $\mathcal{E}$ and an initial density matrix ρ_{init} induce a selective quantum process $\{R_1, R_2, \ldots\}$ as follows: for each n and each sequence $(\tau_1, \ldots, \tau_n) \in \Delta^n$, the probability that $R_1, \ldots, R_n$ take values $\tau_1, \ldots, \tau_n$ is the probability that, if the selective quantum operation $\mathcal{E}$ is iterated n times given initial state ρ_{init}, the resulting classical outputs will be $\tau_1, \ldots, \tau_n$. The probability that R_n takes a particular value τ_n depends on the values taken by $R_1, \ldots, R_{n-1}$ in the following way:

$$\Pr\left[R_n = \tau_n | R_1 = \tau_1, \ldots, R_{n-1} = \tau_{n-1}\right] = p_{\tau_n}\left(\Phi_{\tau_{n-1}} \circ \cdots \circ \Phi_{\tau_1}(\rho_{init})\right).$$

Equivalently, we have

$$\Pr[R_1 = \tau_1, \ldots, R_n = \tau_n] = \operatorname{Tr}(\Psi_{\tau_n} \circ \cdots \circ \Psi_{\tau_1}(\rho_{init})).$$

2.3 Space-bounded quantum Turing machines

In this section we define a variant of the quantum Turing machine model that allows measurements and other non-unitary quantum operations to be performed during a computation, and is appropriate for studying space-bounded computation. This model was described informally in Section 1. The model we define is different from previously defined quantum Turing machines, and is perhaps more accurately viewed as a hybrid between the quantum Turing machine model and the quantum circuit model. The main purpose of defining this model is to illustrate how the main technical results of this paper concerning simulation of selective quantum processes apply to a specific machine model.

A quantum Turing machine will essentially be a classical, deterministic Turing machine (having a read-only input tape and a read/write work tape), along with a quantum tape and a finite internal quantum register. The quantum tape squares each store one qubit. (One could of course consider higher-dimensional quantum systems instead of qubits, but for simplicity we will restrict the quantum tape to contain qubits.) In addition to having a classical internal state, a quantum Turing machine will have a special measurement register, which is to be viewed as being classical—this register will store results of measurements of the quantum part of the machine.

Each step of a quantum Turing machine will consist of two phases: a quantum phase and a classical phase. In the quantum phase, some selective quantum operation is applied to the qubits in the internal quantum register together with the qubit being scanned on the quantum tape. This selective quantum operation will produce a classical output that is written to the measurement register, with the previous contents being overwritten. The second phase is the classical phase, which functions essentially as for an ordinary deterministic Turing machine. The contents of the measurement register may influence the machine's behavior in the classical phase. All movement of tape heads occurs in the classical phase, so the positions of all tape heads, including the quantum tape head, are classical.

More formally, a quantum Turing machine M is specified as follows. First, M has a finite set of internal (classical) states Q, an input tape alphabet Σ, and a work tape alphabet Γ. In addition there is some set Δ of possible states of the measurement register, with the elements of Δ coinciding with the possible outputs of the selective quantum operations performed during the computation. As in the case of classical Turing machines, there are two halting states q_{acc} and q_{rej} contained in Q, as well as an initial state. The quantum phase of each step is determined by a collection

$$\{\mathcal{E}_q \,|\, q \in Q\}$$

of selective quantum operations indexed by the states of M, with the particular operation performed on a given step being the one indexed by the internal state q of the machine at that instant. Each operation $\mathcal{E}_q$ acts on $K+1$ qubits, where K is some fixed number representing the number of qubits in the internal quantum register. As stated above, the output alphabet of all of these operations is Δ. The classical phase of each step is determined by a transition function δ having the form

$$\delta : (Q\backslash\{q_{acc}, q_{rej}\}) \times (\Sigma \cup \{\#\}) \times (\Gamma \cup \{\#\}) \times \Delta \rightarrow Q \times (\Gamma \cup \{\#\}) \times \{-1, 0, +1\}^3$$

The interpretation of the transition function is as follows. Suppose at some instant the internal state of M is q, the symbols being scanned on the input tape and the work tape are $\sigma \in \Sigma$ and $\pi \in \Gamma$, respectively, and the measurement register contains $\tau \in \Delta$. Let

$$(q', \pi', d_1, d_2, d_3) = \delta(q, \sigma, \pi, \tau).$$

Then the machine executes the following actions: (i) change the internal state to q', (ii) write π' on the work tape, and (iii) move the input tape head, work tape head, and qubit tape head according to d_1, d_2, and d_3, respectively. Thus, the classical phase works just like an ordinary deterministic Turing machine, except that the contents of the measurement register can influence the action taken.

The initial configuration of a quantum Turing machine is as follows. The input tape head is scanning the blank tape square immediately to the left of the first input symbol, the work tape contains all blanks, and each qubit on the qubit tape is in the zero state. (Because the work tape and qubit tape are two-way infinite, the starting location of the heads on these tapes is relative and need not be specified.) The internal state of the machine is the initial state, and similar to the quantum tape the K qubits in the internal quantum register are set to the zero state. The symbol in the measurement register is some arbitrarily chosen element of Δ. (This choice does not affect the computation, as this symbol is immediately overwritten by the first quantum operation. However, we wish to have a well-defined initial configuration of any quantum Turing machine, which includes the contents of the measurement register, so it is necessary to make this arbitrary choice.)

The space required for a quantum Turing machine computation is defined similarly to the classical case. (The quantum tape head location is classical, and therefore requires no special consideration when defining space-usage.) A quantum Turing machine M runs in space s if the following holds for every input string $x \in \Sigma^*$:

1. The input tape head never leaves the region contained inclusively between the tape square immediately preceding the first input symbol and the tape square immediately following the last input symbol.

2. The work tape head never moves to the left of its initial position and moves fewer than $s(|x|)$ squares to the right of its initial location.

3. The quantum tape head never moves to the left of its initial position and moves fewer than $s(|x|)$ squares to the right of its initial location.

Definition 9. A language $A \subseteq \Sigma^*$ is in $\mathrm{PrQSPACE}_{\mathbb{A}}(s)$ if and only if there exists a quantum Turing machine M running in space $O(s)$ that satisfies the following conditions: (i) for each $x \in \Sigma^*$, M accepts x with probability exceeding $1/2$ if and only if $x \in A$, and (ii) all selective quantum operations of M are algebraic (i.e., have operator-sum representations consisting of matrices of algebraic numbers).

3 GapL approximable numbers

In this section we define a class of numbers that can be efficiently approximated by ratios of GapL functions, and show that this class includes all real algebraic numbers.

Definition 10. Let $\alpha \in \mathbb{R}$. We say α is *GapL approximable* if there exist $f, g \in \mathrm{GapL}$ such that for all $n \geq 0$ we have $g(1^n) \neq 0$ and

$$\left| \frac{f(1^n)}{g(1^n)} - \alpha \right| < 2^{-n}.$$

Theorem 11. *Let α be any real algebraic number. Then α is GapL approximable.*

The main idea of the proof of this theorem is to use Newton's Method. The proof relies on the following lemma.

Lemma 12. *Let u_0 and u_1 be bivariate integer polynomials and let a_0 and a_1 be integers. Then there exists $f \in \mathrm{GapL}$ such that*

$$f(1^n, c) = \begin{cases} u_c\left(f\left(1^{\lceil n/2 \rceil}, 0\right), f\left(1^{\lceil n/2 \rceil}, 1\right)\right) & n \geq 2 \\ a_c & n = 1 \end{cases}$$

for all $n \geq 0$ and $c \in \{0, 1\}$.

Proof. We will define a logarithmic space nondeterministic Turing machine M_f such that

$$f(1^n, c) = \#M_f(1^n, c) - \#\overline{M_f}(1^n, c)$$

satisfies the recurrence in the statement of the lemma.

Write
$$u_c(x,y) = \sum_{0 \le i,j \le d} u_{c,i,j} x^i y^j.$$
To simplify the presentation of M_f, we define M_c and $M_{c,i,j}$ for $c \in \{0,1\}$, $0 \le i,j \le d$ to be nondeterministic Turing machines that take no input and satisfy
$$\#M_c - \#\overline{M_c} = a_c \quad \text{and} \quad \#M_{c,i,j} - \#\overline{M_{c,i,j}} = u_{c,i,j}.$$
The execution of M_f is as follows.

Description of M_f

```
Input: $(1^n, c)$.

1.  $k \leftarrow \lceil \log n \rceil$.
2.  $z \leftarrow 1$.
3.  Call procedure $P$ on input $c$.
4.  If $z = 1$ then accept, otherwise reject.
```

```
Procedure P

Input: $c$.

5.  If $k = 0$ then
6.      Simulate $M_c$. If $M_c$ rejects then $z \leftarrow -z$.
7.  else
8.      Guess $i, j \in \{0, \ldots, d\}$.
9.      Simulate $M_{c,i,j}$. If $M_{c,i,j}$ rejects then $z \leftarrow -z$.
10.     $k \leftarrow k - 1$.
11.     Repeat $i$ times: Call procedure $P$ on input 0.
12.     Repeat $j$ times: Call procedure $P$ on input 1.
13.     $k \leftarrow k + 1$.
```

The variables k and z are global, while i, j, and any auxiliary variables needed by procedure P are local. As i, j, and all required auxiliary variables are constant in size, M_f will need to store only a constant amount of information for each level of the recursion. As the recursion will have depth at most logarithmic in n, M_f requires space $O(\log n)$ to implement the recursion. Because each of the machines M_c and $M_{c,i,j}$ require only constant space, it follows that M_f may be taken to run in space $O(\log n)$.

Now let us analyze the computation of M_f. Each execution of procedure P causes the computation of M_f to branch along several computation paths, each path having the effect of either leaving z unchanged or replacing z with $-z$. Let $r^+(k,c)$ denote the number of computation paths induced by calling P on input c and global value k that leave z unchanged, let $r^-(k,c)$ denote the number of computation paths induced by calling P on input c and global value k that result in z being replaced by $-z$, and define $r(k,c) = r^+(k,c) - r^-(k,c)$. Note that we have
$$\#M_f(1^n, c) - \#\overline{M_f}(1^n, c) = r(\lceil \log n \rceil, c).$$
As $\lceil \log \lceil n/2 \rceil \rceil = \lceil \log n \rceil - 1$ for any integer $n \ge 2$, it remains to prove that the function r obeys the recurrence
$$r(k,c) = \begin{cases} u_c(r(k-1,0), r(k-1,1)) & k \ge 1 \\ a_c & k = 0. \end{cases}$$

In case $k = 0$, procedure P on input c induces $\#M_c$ computation paths that do not modify z and $\#\overline{M_c}$ paths that replace z with $-z$, and thus $r(0,c) = a_c$. Now suppose $k \geq 1$ and assume as an inductive hypothesis that the number of paths induced by P on input b that do not change z (replace z with $-z$) when k is replaced by $k-1$ is described by $r^+(k-1,b)$ ($r^-(k-1,b)$, respectively) for each $b \in \{0,1\}$. For each pair i, j that may be guessed (on line number 8), it follows from the binomial theorem that the number of computation paths induced by the remaining portion of $P(c)$ that have the effect of leaving z unchanged minus the number of paths that replace z by $-z$ is given by $u_{c,i,j}\, r(k-1,0)^i\, r(k-1,1)^j$. We therefore conclude that

$$r(k,c) \;=\; \sum_{i,j} u_{c,i,j}\, r(k-1,0)^i\, r(k-1,1)^j \;=\; u_c(r(k-1,0), r(k-1,1))$$

for $k \geq 1$, which completes the proof. $\square$

Proof. Proof of Theorem 11 Clearly 0 is GapL approximable, so consider the case $\alpha \neq 0$. Let

$$p(x) = p_d x^d + \cdots + p_0$$

be an integer polynomial such that $p(\alpha) = 0$, and assume without loss of generality that α is not a zero of the derivative of p, i.e., $p'(\alpha) \neq 0$.

Lemma 12 will allow us to use Newton's Method to approximate α by GapL functions. We have that there exist positive constants ξ and K, where ξ and ξK are at most $1/2$, such that for $x_0 \in (\alpha - \xi, \alpha + \xi)$ and

$$x_{k+1} = x_k - \frac{p(x_k)}{p'(x_k)},$$

the inequality

$$|x_{k+1} - \alpha| \leq K |x_k - \alpha|^2$$

is satisfied for all $k \geq 0$. Thus we have

$$|x_k - \alpha| < 2^{-2^k}$$

for every $k \geq 0$.

Define

$$\begin{aligned} u_0(x,y) &= \sum_{j=0}^{d} (j-1) p_j x^j y^{d-j} \\ u_1(x,y) &= \sum_{j=1}^{d} j p_j x^{j-1} y^{d-j+1}, \end{aligned}$$

and note that

$$\frac{u_0(x,y)}{u_1(x,y)} = \frac{x}{y} - \frac{p(x/y)}{p'(x/y)}.$$

Let $a_0, a_1 \in \mathbb{Z}$, $a_1 \neq 0$, be such that $|\alpha - a_0/a_1| < \xi$. By Lemma 12 there exists $f \in$ GapL such that

$$\frac{f(1^n,0)}{f(1^n,1)} = \frac{u_0(f(1^{\lceil n/2 \rceil},0), f(1^{\lceil n/2 \rceil},1))}{u_1(f(1^{\lceil n/2 \rceil},0), f(1^{\lceil n/2 \rceil},1))} = \frac{f(1^{\lceil n/2 \rceil},0)}{f(1^{\lceil n/2 \rceil},1)} - \frac{p\left(\frac{f(1^{\lceil n/2 \rceil},0)}{f(1^{\lceil n/2 \rceil},1)}\right)}{p'\left(\frac{f(1^{\lceil n/2 \rceil},0)}{f(1^{\lceil n/2 \rceil},1)}\right)}$$

for $n \geq 2$, and $f(1,0)/f(1,1) = a_0/a_1$. Consequently

$$\left| \frac{f(1^n,0)}{f(1^n,1)} - \alpha \right| < 2^{-2^{\lceil \log n \rceil}} \leq 2^{-n}$$

for every $n \geq 1$. We may now define $g, h \in \mathrm{GapL}$ that satisfy

$$g(1^n) = \begin{cases} f(1^n,0) & n \geq 1 \\ a_0 & n = 0 \end{cases}$$

and

$$h(1^n) = \begin{cases} f(1^n,1) & n \geq 1 \\ a_0 & n = 0, \end{cases}$$

so that

$$\left| \frac{g(1^n)}{h(1^n)} - \alpha \right| < 2^{-n}$$

for all $n \geq 0$. Therefore we have that α is GapL approximable. □

4 A matrix problem in probabilistic logspace

The purpose of this section is to state and prove Lemma 14, which states that a particular matrix problem can be solved in PL. In the section following this one, this problem is related to the problem of simulating logarithmic-space quantum computations.

We start this section by defining a class of families of matrices whose entries are linear combinations of a finite set of algebraic numbers and where the coefficients are ratios of GapL functions. Lemma 14 concerns such families of matrices, and in addition families of matrices of this type will play an important role in our simulation of space-bounded processes in later sections.

Definition 13. Let p be a positive integer polynomial and for each $x \in \Sigma^*$ let M_x be a $p(|x|) \times p(|x|)$ matrix. The collection of matrices $\{M_x\}$ is said to have a $\mathrm{GapL}_{\mathbb{A}}$-description if there exist GapL functions a and b and a finite collection $\{\alpha_1, \ldots, \alpha_n\}$ of algebraic numbers such that

$$M_x[i,j] = \sum_{l=1}^{n} \frac{a(x,i,j,l)}{b(x,i,j,l)} \alpha_l$$

for every $x \in \Sigma^*$ and $1 \leq i, j \leq p(|x|)$.

The elements of a family of matrices $\{M_x\}$ having a $\mathrm{GapL}_{\mathbb{A}}$-description may, in general, have complex entries. In this section, however, we will be working just over the real numbers.

Lemma 14. *Let p be a positive integer polynomial and assume, for each $x \in \Sigma^*$, M_x is a $p(|x|) \times p(|x|)$ real matrix having eigenvalues bounded by 1 in absolute value for which the series $\sum_{t \geq 0} M_x^t[p(|x|), 1]$ converges. If the collection $\{M_x\}$ has a* $\mathrm{GapL}_{\mathbb{A}}$*-description, then the language*

$$\left\{ x \in \Sigma^* \; : \; \sum_{t \geq 0} M_x^t[p(|x|), 1] > 0 \right\}$$

is in PL.

Before giving the proof of Lemma 14, we will outline briefly the main idea of the proof. Our goal will be to define a GapL function $F(x)$ such that

$$F(x) > 0 \quad \Leftrightarrow \quad \sum_{t \geq 0} M_x^t[p(|x|), 1] > 0. \tag{2}$$

By Theorem 5, this will imply that the language in the statement of the lemma is in PL.

Under the assumption that M_x has eigenvalues bounded in absolute value by 1 and the series $\sum_{t \geq 0} M_x^t[p(|x|), 1]$ converges, we have

$$\begin{aligned} \sum_{t \geq 0} M_x^t[p(|x|), 1] &= \lim_{z \to 1} (I - zM_x)^{-1}[p(|x|), 1] \\ &= (-1)^{1+p(|x|)} \lim_{z \to 1} \frac{\det((I - zM_x)_{(1, p(|x|))})}{\det(I - zM_x)}. \end{aligned} \tag{3}$$

Here, the notation $(I - zM_x)_{(1,p(|x|))}$ means the $(p(|x|) - 1) \times (p(|x|) - 1)$ matrix obtained by removing row number 1 and column number $p(|x|)$ from $I - zM_x$.

Equation 3 will allow us to define a GapL function F for which 2 holds. The principle facts used to construct such a function F are (i) that the entries of M_x can be approximated by ratios of GapL functions, following from the fact that $\{M_x\}$ has a $\mathrm{GapL}_{\mathbb{A}}$-description along with the fact that the algebraic numbers can be approximated by ratios of GapL functions; (ii) the determinants can be handled using Theorem 8; and (iii) the limit can be approximated by substituting for z a value very close to 1.

Although this construction is simple in principle, the details of the construction are somewhat technical. The main difficulty is that a correct determination of the sign of $\sum_{t \geq 0} M_x^t[p(|x|), 1]$ will require a sufficiently accurate approximation. Because we are working in the unbounded error setting, the only source we have for obtaining bounds on the required accuracy are properties of algebraic numbers. This is the purpose of Lemma 16 stated below. Informally, when this lemma is used in the proof of Lemma 14, it will imply two things: (i) the quantity $\sum_{t \geq 0} M_x^t[p(|x|), 1]$ cannot be "too close" to zero if it is nonzero, and (ii) the values substituted for z and α do not need to be "too close" to 1 and the true value of α, respectively, to get a sufficiently accurate approximation to determine the sign of $\sum_{t \geq 0} M_x^t[p(|x|), 1]$.

Now let us move on to the actual proof of Lemma 14, which occupies the remainder of this section. The proof will require a definition and a few technical lemmas, which follow. (It should be mentioned that no attempt has been made to optimize the parameters in these lemmas or their proofs. Instead we have chosen the various parameters in a way that makes the required arithmetic as simple and transparent as possible.)

Definition 15. For any univariate polynomial $u(x) = \sum_j u_j x^j$ or bivariate polynomial $u(x, y) = \sum_{i,j} u_{i,j} x^i y^j$ we define $\|u\|$ to be the largest coefficient of u in absolute value, i.e., $\max_j\{|u_j|\}$ or $\max_{i,j}\{|u_{i,j}|\}$, respectively.

Lemma 16. *For any real algebraic number α there exist positive integer constants C_1 and C_2 such that the following holds. For all $N \geq 1$, if u and v are bivariate integer polynomials such that* $\deg(u), \deg(v) \leq N$, *$\|u\|, \|v\| \leq 2^N$, and $\lim_{z \to 1} \frac{u(\alpha,z)}{v(\alpha,z)}$ exists (taking a finite value), then*

$$\left| \lim_{z \to 1} \frac{u(\alpha, z)}{v(\alpha, z)} - 2^{-C_1 N^2} \right| \geq 2^{-C_1 N^2}.$$

Furthermore, for any real numbers α_0 and z_0 satisfying $|\alpha - \alpha_0| \le 2^{-C_2N^2}$ and $1 - 2^{-C_2N^2} \le z_0 < 1$ we have $v(\alpha_0, z_0) \neq 0$ and
$$\left|\lim_{z\to 1} \frac{u(\alpha,z)}{v(\alpha,z)} - \frac{u(\alpha_0,z_0)}{v(\alpha_0,z_0)}\right| < 2^{-C_1N^2}.$$

Lemma 17. *Let u and v be polynomials and let z and δ satisfy $0 < \delta \le |v(0)|$ and $|z| \le \frac{\delta}{4\,\|v\|}$. Then we have $|v(z)| \ge \delta/2$ and*
$$\left|\frac{u(0)}{v(0)} - \frac{u(z)}{v(z)}\right| \;\le\; \frac{8\,\|u\|\,\|v\|}{\delta^2}|z|.$$

Proof. As $|v(0)| \le \|v\|$ we have $|z| \le 1/4$. Thus,
$$|v(z)| = \left|v_0 + \sum_{j\ge 1} v_j z^j\right| \ge \left||v(0)| - \sum_{j\ge 1} |v_j|\cdot|z|^j\right| \ge |v(0)| - 2\,|z|\,\|v\| \ge \frac{\delta}{2},$$

and
$$\begin{aligned}\left|\frac{u(0)}{v(0)} - \frac{u(z)}{v(z)}\right| &= \frac{|u(0)v(z) - v(0)u(z)|}{|v(0)\,v(z)|}\\ &\le \frac{2}{\delta^2}\left|\sum_{j\ge1}(u_0v_j - v_0u_j)z^j\right|\\ &\le \frac{8\,\|u\|\,\|v\|}{\delta^2}|z|\end{aligned}$$

as claimed. □

Theorem 18 (Mahler)**.** *Let f and g be integer polynomials of degree d_f and d_g, respectively, and let α satisfy $f(\alpha) = 0$ and $g(\alpha) \neq 0$. Then*
$$|g(\alpha)| \ge \frac{1}{(d_f+d_g-1)!\,\|f\|^{d_g}\,\|g\|^{d_f-1}\,(|\alpha|^{d_f-1}+\cdots+|\alpha|+1)}.$$

Proof. A proof can be found on pages 44–46 of [Mah61]. □

Proof of Lemma 16. We may write
$$u(\alpha, 1-z) = \sum_{j=0}^{N} u_j(\alpha)z^j \quad\text{ and }\quad v(\alpha,1-z) = \sum_{j=0}^{N} v_j(\alpha)z^j$$

for integer polynomials u_j and v_j, $0 \le j \le N$, satisfying $\deg(u_j), \deg(v_j) \le N$ and $\|u_j\|, \|v_j\| \le 2^{2N}$. For $0 \le j \le N$ we have
$$|u_j(\alpha)| \le \|u_j\| \sum_{i=0}^{N} |\alpha|^i \le 2^{2N}(1+|\alpha|)^N,$$

and similarly $|v_j(\alpha)| \le 2^{2N}(1+|\alpha|)^N$.

The assumption that $\lim_{z\to 1} \frac{u(\alpha,z)}{v(\alpha,z)}$ exists implies that $v(\alpha, z)$ is not identically zero for all values of z, so there must be some choice of j for which $v_j(\alpha) \neq 0$. Fix $k = \min\{j : v_j(\alpha) \neq 0\}$. As the limit takes a finite value, we conclude that $u_0(\alpha) = \cdots = u_{k-1}(\alpha) = 0$, and that

$$\lim_{z\to 1} \frac{u(\alpha,z)}{v(\alpha,z)} = \frac{u_k(\alpha)}{v_k(\alpha)}.$$

By taking f to be any integer polynomial of which α is a zero in Theorem 18, we may conclude that there exists a positive constant C_3 (depending on α) such that $|u_j(\alpha)| \geq 2^{-C_3N^2}$ whenever $u_j(\alpha) \neq 0$, and similarly, $|v_j(\alpha)| \geq 2^{-C_3N^2}$ whenever $v_j(\alpha) \neq 0$. Without loss of generality we may assume that C_3 is an integer with $C_3 \geq 3 + \log_2(1 + |\alpha|)$. Thus

$$\left|\frac{u_k(\alpha)}{v_k(\alpha)}\right| \geq \frac{2^{-C_3N^2}}{2^{2N}\,(1+|\alpha|)^N} \geq 2^{-2\,C_3N^2}$$

whenever $u_k(\alpha) \neq 0$. Define $C_1 = 2\,C_3 + 1$. As $2^{-C_1N^2} \leq \frac{1}{2}2^{-2C_3N^2}$, we have

$$\left|\frac{u_k(\alpha)}{v_k(\alpha)} - 2^{-C_1N^2}\right| \geq 2^{-C_1N^2},$$

which proves the first part of the lemma.

Now we move on to the second part of the lemma. Choosing $C_2 = 6\,C_3 + 7$ will suffice, as we now show. Define

$$a(z) = z^{-k}u(\alpha, 1-z) \quad \text{ and } \quad b(z) = z^{-k}v(\alpha, 1-z).$$

From the discussion above, a and b are integer polynomials,

$$\lim_{z\to 1} \frac{u(\alpha,z)}{v(\alpha,z)} = \frac{a(0)}{b(0)}$$

and

$$\frac{u(\alpha,1-z)}{v(\alpha,1-z)} = \frac{a(z)}{b(z)}$$

for all values of z for which $v(\alpha, 1-z) \neq 0$. Note that $\deg(a)$, $\deg(b) \leq N$, $\|a\|, \|b\| \leq 2^{C_3N}$, and $|b(0)| = |v_k(\alpha)| \geq 2^{-C_3N^2}$. Applying Lemma 17 with $\delta = 2^{-C_3N^2}$, we conclude that if $0 < \xi \leq 2^{-(2C_3+2)N^2}$, then $|b(\xi)| \geq \frac{1}{2}2^{-C_3N^2}$ and

$$\left|\frac{a(0)}{b(0)} - \frac{a(\xi)}{b(\xi)}\right| \leq 2^{(4C_3+3)N^2}\xi.$$

It follows that if $\xi \leq 2^{-C_2N^2}$, then

$$\left|\frac{a(0)}{b(0)} - \frac{a(\xi)}{b(\xi)}\right| < \frac{1}{2}2^{-C_1N^2}.$$

Now suppose that $\xi \leq 2^{-C_2N^2}$ has been fixed. Define

$$c(y) = u(\alpha + y, 1-\xi) \quad \text{ and } \quad d(y) = v(\alpha + y, 1-\xi).$$

Then $\deg(c)$, $\deg(d) \leq N$ and $\|c\|$, $\|d\| \leq (N+1)2^{2N}(1+|\alpha|)^N \leq 2^{C_3 N}$. Noting that $|d(0)| = |v(\alpha, 1-\xi)| = |b(\xi)| \geq \frac{1}{2}2^{-C_3 N^2}$, we may again apply Lemma 17, this time with $\delta = \frac{1}{2}2^{-C_3 N^2}$. If η is any real number with $|\eta| \leq 2^{-(2C_3+3)N^2}$, then $d(\eta) \neq 0$ and

$$\left| \frac{c(0)}{d(0)} - \frac{c(\eta)}{d(\eta)} \right| \leq 2^{(4C_3+5)N^2} |\eta|.$$

Consequently, if $|\eta| \leq 2^{-C_2 N^2}$ then

$$\left| \frac{c(0)}{d(0)} - \frac{c(\eta)}{d(\eta)} \right| \leq \frac{1}{2} 2^{-C_1 N^2}.$$

At this point we have

$$\begin{aligned}\left| \lim_{z\to 1} \frac{u(\alpha,z)}{v(\alpha,z)} - \frac{u(\alpha+\eta, 1-\xi)}{v(\alpha+\eta,1-\xi)} \right| &\leq \left| \lim_{z\to 1} \frac{u(\alpha,z)}{v(\alpha,z)} - \frac{u(\alpha, 1-\xi)}{v(\alpha,1-\xi)} \right| + \left| \frac{u(\alpha, 1-\xi)}{v(\alpha,1-\xi)} - \frac{u(\alpha+\eta, 1-\xi)}{v(\alpha+\eta,1-\xi)} \right| \\ &= \left| \frac{a(0)}{b(0)} - \frac{a(\xi)}{b(\xi)} \right| + \left| \frac{c(0)}{d(0)} - \frac{c(\eta)}{d(\eta)} \right| \\ &\leq 2^{-C_1 N^2},\end{aligned}$$

which completes the proof. □

Proof. Proof of Lemma 14 Let $\{\alpha_1, \ldots, \alpha_n\}$ be a finite collection of algebraic numbers and let a and b be GapL functions such that

$$M_x[i,j] = \sum_{l=1}^{n} \frac{a(x,i,j,l)}{b(x,i,j,l)} \alpha_l,$$

for every $x \in \Sigma^*$ and $1 \leq i,j \leq p(|x|)$. Because each matrix M_x is real, we may assume without loss of generality that $\alpha_1, \ldots, \alpha_n$ are real, for otherwise we may simply take the real part of each of these numbers.

For the given collection $\{\alpha_1, \ldots, \alpha_n\}$ of algebraic numbers, there exists a single real algebraic number α, an integer k, and integer polynomials $\varphi_1, \ldots, \varphi_n$ such that

$$\alpha_l = \frac{\varphi_l(\alpha)}{k},$$

for $1 \leq l \leq n$. This is because the field $\mathbb{Q}[\alpha_1, \ldots, \alpha_n]$ is a finite degree separable extension of the rationals, and therefore $\mathbb{Q}[\alpha] = \mathbb{Q}[\alpha_1, \ldots, \alpha_n]$ for some algebraic number α—see, for example, [Isa94] for a proof of this fact. Letting d be the degree of the minimal polynomial of α, it may be assumed that $\deg(\varphi_l) \leq d$ for $1 \leq l \leq n$. Define bivariate integer polynomials $w_1, \ldots, w_n$ as

$$w_l(y,z) = z^d \, \varphi_l(y/z)$$

for $1 \leq l \leq n$. It follows that $\deg(w_l) \leq d$. Note that d, k, $\varphi_1, \ldots, \varphi_n$ and $w_1, \ldots, w_n$ are independent of the input x.

Define

$$D(x) = k \prod_{i=1}^{p(|x|)} \prod_{j=1}^{p(|x|)} \prod_{l=1}^{n} b(x,i,j,l).$$

By Theorem 6 we may conclude that $D \in \mathrm{GapL}$. Let $A_x(y)$ be a $p(|x|) \times p(|x|)$ matrix defined by

$$A_x(y)[i,j] \;=\; D(x) \sum_{l=1}^{n} \frac{a(x,i,j,l)}{b(x,i,j,l)} \, \frac{\varphi_l(y)}{k}.$$

At this point y is a variable that will momentarily take the value of an approximation to α. Notice that $A_x(\alpha) = D(x)M_x$, and that $A_x(y)[i,j]$ is an integer polynomial in y for each $x \in \Sigma^*$ and $1 \leq i,j \leq p(|x|)$.

Next, for each $x \in \Sigma^*$ define bivariate integer polynomials u_x and v_x as follows:

$$\begin{aligned} u_x(y,z) &= (-1)^{1+p(|x|)}\, D(x)\, \det((D(x)\, I - z\, A_x(y))_{(1,p(|x|))}), \\ v_x(y,z) &= \det(D(x)\, I - z\, A_x(y)). \end{aligned}$$

It follows from the fact that $D(x)I - zA_x(y)$ is a matrix of polynomial size whose entries are constant-degree polynomials that there exists a positive integer polynomial N such that

$$\deg(u_x), \deg(v_x) \leq N(|x|)$$

and $\|u_x\|, \|v_x\| \leq 2^{N(|x|)}$ for all $x \in \Sigma^*$. Given that M_x has eigenvalues bounded by 1 in absolute value and the sum $\sum_{t\geq 0} M_x^t[p(|x|),1]$ converges for each x, we have

$$\lim_{z\to 1} \frac{u_x(\alpha,z)}{v_x(\alpha,z)} = \lim_{z\to 1} (I - zM_x)^{-1}[p(|x|),1] \;=\; \sum_{t\geq 0} M_x^t[p(|x|),1].$$

By Lemma 16, there exist positive integer constants C_1 and C_2 such that

$$v_x\left(\alpha_0, 1 - 2^{-C_2 N(|x|)^2}\right) \neq 0,$$

$$\left| \sum_{t\geq 0} M_x^t[p(|x|),1] - 2^{-C_1 N(|x|)^2} \right| \;\geq\; 2^{-C_1 N(|x|)^2}, \tag{4}$$

and

$$\left| \sum_{t\geq 0} M_x^t[p(|x|),1] - \frac{u_x\left(\alpha_0, 1 - 2^{-C_2 N(|x|)^2}\right)}{v_x\left(\alpha_0, 1 - 2^{-C_2 N(|x|)^2}\right)} \right| \;<\; 2^{-C_1 N(|x|)^2} \tag{5}$$

whenever $|\alpha_0 - \alpha| < 2^{-C_2 N(|x|)^2}$. Define $\nu(|x|) = C_2 N(|x|)^2$. By Theorem 6 and Theorem 11 there exist GapL functions f and g such that

$$\left| \frac{f(1^{\nu(|x|)})}{g(1^{\nu(|x|)})} - \alpha \right| < 2^{-\nu(|x|)}$$

for every $x \in \Sigma^*$.

Next, define

$$h(x,i,j) = \sum_{l=1}^{n} \left(\prod_{i'\neq i} \prod_{j'\neq j} \prod_{l'\neq l} b(x,i',j',l') \right) a(x,i,j,l)\, w_l \left(f\left(1^{\nu(|x|)}\right), g\left(1^{\nu(|x|)}\right) \right)$$

for $1 \leq i,j \leq p(|x|)$. Here, the products involving i' and j' are over all values in the range $\{1,\ldots,p(|x|)\}$ not equal to i and j, respectively, and the product involving l' is over all values in the range $\{1,\ldots,n\}$ not equal to l. Let H_x denote the $p(|x|)\times p(|x|)$ matrix defined by $H_x[i,j] = h(x,i,j)$. We have that

$$H_x = \left(g\left(1^{\nu(|x|)}\right)\right)^d A_x\left(\frac{f(1^{\nu(|x|)})}{g(1^{\nu(|x|)})}\right).$$

By Theorem 6, we may conclude that $h \in \mathrm{GapL}$. Continuing on, define

$$r(x,i,j) \;=\; D(x)\left(g\left(1^{\nu(|x|)}\right)\right)^d 2^{\nu(|x|)} I[i,j] - \left(2^{\nu(|x|)}-1\right)h(x,i,j),$$

and let R_x be the $p(|x|)\times p(|x|)$ matrix defined by $R_x[i,j] = r(x,i,j)$. We have

$$R_x \;=\; D(x)\left(g\left(1^{\nu(|x|)}\right)\right)^d 2^{\nu(|x|)} I - (2^{\nu(|x|)}-1)H_x.$$

By Theorem 6, $r \in \mathrm{GapL}$. Next, define

$$\begin{aligned} U(x) &= (-1)^{p(|x|)+1} D(x)\left(g\left(1^{\nu(|x|)}\right)\right)^d 2^{\nu(|x|)}\det((R_x)_{(1,p(|x|))}),\\ V(x) &= \det(R_x). \end{aligned}$$

By Theorem 6 and Theorem 8, $U, V \in \mathrm{GapL}$. Finally, define

$$F(x) \;=\; 2^{C_1 N(|x|)^2} U(x)V(x) - (V(x))^2.$$

By Theorem 6, $F \in \mathrm{GapL}$.

It remains to show that for every $x \in \Sigma^*$ (with $|x| \geq 2$), $F(x) > 0$ if and only if

$$\sum_{t\geq 0} M_x^t[p(|x|),1] > 0.$$

It may be verified that

$$U(x) \;=\; \left(\left(g\left(1^{\nu(|x|)}\right)\right)^d 2^{\nu(|x|)}\right)^{p(|x|)} u_x\left(\frac{f(1^{\nu(|x|)})}{g(1^{\nu(|x|)})}, 1-2^{\nu(|x|)}\right)$$

and

$$V(x) \;=\; \left(\left(g\left(1^{\nu(|x|)}\right)\right)^d 2^{\nu(|x|)}\right)^{p(|x|)} v_x\left(\frac{f(1^{\nu(|x|)})}{g(1^{\nu(|x|)})}, 1-2^{\nu(|x|)}\right).$$

Thus we have $V(x) \neq 0$. Furthermore $F(x) > 0$ if and only if

$$\frac{U(x)}{V(x)} > 2^{-C_1 N(|x|)^2},$$

which is equivalent to

$$\frac{u_x\left(\widetilde{\alpha}, 1-2^{\nu(|x|)}\right)}{v_x\left(\widetilde{\alpha}, 1-2^{\nu(|x|)}\right)} \;>\; 2^{-C_1 N(|x|)^2}. \tag{6}$$

By 4 and 5, the inequality 6 holds if and only if

$$\sum_{t\geq 0} M_x^t[p(|x|),1] > 0,$$

as required. □

5 Simulation of quantum processes

In the previous section it was shown that a particular matrix problem can be solved in probabilistic logarithmic space. The purpose of this section is to demonstrate that this fact can be used to solve a related problem concerning selective quantum processes.

Definition 13 introduced the notion of a $\mathrm{GapL}_{\mathbb{A}}$-description of a family of matrices. A similar notion for selective quantum operations is described in the next definition.

Definition 19. Let p be a positive integer polynomial and for each $x \in \Sigma^*$ let $\mathcal{E}_x$ be a selective quantum operation acting on $p(|x|) \times p(|x|)$ density matrices and having classical output set Δ. The collection of operations $\{\mathcal{E}_x\}$ is said to have a $\mathrm{GapL}_{\mathbb{A}}$-description if there exists a positive integer polynomial m, GapL functions a and b, and a finite collection $\{\alpha_1, \ldots, \alpha_n\}$ of algebraic numbers such that each operation $\mathcal{E}_x$ is described by the $p(|x|) \times p(|x|)$ matrices

$$\{A_{\tau,k} \,:\, \tau \in \Delta,\, 1 \leq k \leq m(|x|)\}$$

given by

$$A_{\tau,k}[i,j] \;=\; \sum_{l=1}^{n} \frac{a(x,\tau,k,i,j,l)}{b(x,\tau,k,i,j,l)}\,\alpha_l$$

for every $x \in \Sigma^*$, $\tau \in \Delta$, $1 \leq k \leq m(|x|)$, and $1 \leq i,j \leq p(|x|)$.

Theorem 20. *Let p be a positive integer polynomial and for each $x \in \Sigma^*$ let $\mathcal{E}_x$ be a selective quantum operation acting on $p(|x|) \times p(|x|)$ density matrices and having output set Δ, and let ρ_x be a $p(|x|) \times p(|x|)$ density matrix. Assume the collection of operations $\{\mathcal{E}_x\}$ and the collection of density matrices $\{\rho_x\}$ have $\mathrm{GapL}_{\mathbb{A}}$-descriptions. Let $\{R_{x,1}, R_{x,2}, \ldots\}$ be the selective quantum process induced by $\mathcal{E}_x$ and ρ_x. Then for any $\tau_0 \in \Delta$ and any real algebraic number β, the language $\{x \in \Sigma^* \,:\, \Pr[\exists\, t \,:\, R_{x,t} = \tau_0] > \beta\}$ is in* PL.

This theorem implies that a wide range of logarithmic-space quantum computations can be simulated classically in the unbounded error setting. It will be demonstrated in the next section how this theorem applies to the particular quantum Turing machine model defined in Section 2.3.

The first step in proving Theorem 20 will be to note that the selective quantum operations and density matrices underlying a given selective quantum process may be assumed to be real.

Proposition 21. *Let $\mathcal{E}$ be a selective quantum operation described by matrices $\{A_{\tau,k}\} \subseteq \mathbb{C}^{N\times N}$, let $\rho \in \mathbb{C}^{N\times N}$ be a density matrix, and let $\{R_1, R_2, \ldots\}$ be the selective quantum process induced by $\mathcal{E}$ and initial state ρ. Define real matrices $\{B_{\tau,k}\} \subseteq \mathbb{R}^{2N\times 2N}$ and $\xi \in \mathbb{R}^{2N\times 2N}$ as follows:*

$$\begin{aligned}
B_{\tau,k}[2i-1,2j-1] &= \Re(A_{\tau,k}[i,j])\\
B_{\tau,k}[2i-1,2j] &= \Im(A_{\tau,k}[i,j])\\
B_{\tau,k}[2i,2j-1] &= -\Im(A_{\tau,k}[i,j])\\
B_{\tau,k}[2i,2j] &= \Re(A_{\tau,k}[i,j])
\end{aligned}$$

and

$$\begin{aligned}
\xi[2i-1,2j-1] &= \tfrac{1}{2}\,\Re(\rho[i,j])\\
\xi[2i-1,2j] &= \tfrac{1}{2}\,\Im(\rho[i,j])\\
\xi[2i,2j-1] &= -\tfrac{1}{2}\,\Im(\rho[i,j])\\
\xi[2i,2j] &= \tfrac{1}{2}\,\Re(\rho[i,j])
\end{aligned}$$

Then $\{B_{\tau,k}\}$ describes a selective quantum operation, ξ is a density matrix, and this pair also induces the selective quantum process $\{R_1, R_2, \ldots\}$.

Proof. This is a straightforward computation. □

Next, recall that for $N \times N$ matrices A and B, the Kronecker product $A \otimes B$ is an $N^2 \times N^2$ matrix determined by the formula

$$(A \otimes B)[(i_0 - 1)N + i_1, (j_0 - 1)N + j_1] = A[i_0, j_0]\, B[i_1, j_1]$$

for $1 \leq i_0, i_1, j_0, j_1 \leq N$. Also, the mapping *vec* from $N \times N$ matrices to N^2 dimensional (column) vectors is defined as follows:

$$vec(A)[(i-1)N + j] = A[i, j]$$

for $1 \leq i, j \leq N$. One may view $vec(A)$ as the column vector obtained by stacking the rows of A on top of one another. Obviously *vec* is a linear mapping. For $N \times N$ matrices A, B, and C we have

$$vec(ABC) = (A \otimes C^{\mathsf{T}})vec(B)$$

and

$$\operatorname{Tr}\left(A^{\mathsf{T}} B\right) = vec(A)^{\mathsf{T}} vec(B).$$

The Kronecker product and the *vec* mapping allow one to describe selective quantum processes in a form that more closely resembles an ordinary Markov chain (although the underlying matrix will of course not be a stochastic matrix in this case). The following lemma gives such a description that is convenient for proving Theorem 20.

Lemma 22. *Let $\{A_{\tau,k}\} \subseteq \mathbb{C}^{N\times N}$ describe a selective quantum operation with output set Δ, let $\rho \in \mathbb{C}^{N\times N}$ be a density matrix, and let $\{R_1, R_2, \ldots\}$ be the induced selective quantum process. Let β be any complex number, let $\tau_0 \in \Delta$, and define an $(N^2+2) \times (N^2+2)$ matrix M as follows:*

$$M = \begin{pmatrix} 0 & 0 & 0 \\ vec(\rho) & \displaystyle\sum_{\tau\neq\tau_0} \sum_k A_{\tau,k} \otimes \overline{A_{\tau,k}} & 0 \\ -\beta & vec\left(\displaystyle\sum_k A^*_{\tau_0,k} A_{\tau_0,k}\right)^{\mathsf{T}} & 0 \end{pmatrix}$$

Then all eigenvalues of M are bounded in absolute value by 1. Furthermore, we have $M[N^2+2, 1] = -\beta$ and

$$M^{t+1}[N^2+2, 1] = \Pr[R_1 \neq \tau_0, \ldots, R_{t-1} \neq \tau_0, R_t = \tau_0]$$

for every $t \geq 1$.

The following lemma will be used in the proof of Lemma 22. The lemma can be found in [TD00]; we include a proof for completeness.

Lemma 23. *Let $A_1, \ldots, A_m$ be $N \times N$ matrices such that $\sum_{k=1}^m A_k^* A_k \leq I$ (i.e., such that $I - \sum_{k=1}^m A_k^* A_k$ is positive semidefinite). Then $\sum_{k=1}^m A_k \otimes \overline{A_k}$ has eigenvalues bounded by 1 in absolute value.*

Proof. Define

$$\Psi(X) = \sum_{k=1}^{m} A_k X A_k^*$$

for all $X \in \mathbb{C}^{N\times N}$. The condition $\sum_{k=1}^m A_k^* A_k \leq I$ implies $\operatorname{Tr}(\Psi(X)) \leq \operatorname{Tr}(X)$ whenever X is positive semidefinite.

Suppose that v is an eigenvector of $\sum_{k=1}^{m} A_k \otimes \overline{A_k}$ with eigenvalue λ, and let B be the $N \times N$ matrix with $vec(B) = v$. Then

$$vec(\Psi(B)) = vec\left(\sum_{k=1}^{m} A_k B A_k^*\right) = \left(\sum_{k=1}^{m} A_k \otimes \overline{A_k}\right) vec(B) = \lambda vec(B),$$

which implies that $\Psi(B) = \lambda B$. Because $\Psi(X^*) = \Psi(X)^*$ for every X, we also have $\Psi(B^*) = \overline{\lambda} B^*$.

Now, as $B \neq 0$, there exists a unit vector u such that $u^* B u \neq 0$. Define

$$C = (u^* B^* u)\, B + (u^* B u)\, B^*.$$

Because C is Hermitian, there exists some $\varepsilon > 0$ such that $I + \varepsilon C$ is positive semidefinite.

For any positive integer t the mapping $\Psi^t = \Psi \circ \cdots \circ \Psi$ (t times) is linear, completely positive, and cannot increase trace of positive semidefinite matrices. Thus

$$0 \leq u^* \Psi^t (I + \varepsilon C) u \leq \mathrm{Tr}\left(\Psi^t (I + \varepsilon C)\right) \leq \mathrm{Tr}(I + \varepsilon C).$$

(The important fact is that the upper bound is independent of t.) However,

$$u^* \Psi^t (I + \varepsilon C) u = u^* \Psi^t (I) u + 2\, \varepsilon\, \Re\left(\lambda^t\right) |u^* B u|^2.$$

Because $u^* \Psi^t (I) u \geq 0$ and both ε and $|u^* B u|^2$ are positive, it follows that $|\lambda| \leq 1$. □

Proof. Proof of Lemma 22 Because $\{A_{\tau,k}\}$ describes a selective quantum operation we have

$$\sum_{\tau} \sum_{k} A_{\tau,k}^* A_{\tau,k} = I$$

and thus

$$\sum_{\tau \neq \tau_0} \sum_{k} A_{\tau,k}^* A_{\tau,k} \leq I.$$

Therefore, by Lemma 23, all eigenvalues of

$$\sum_{\tau \neq \tau_0} \sum_{k} A_{\tau,k} \otimes \overline{A_{\tau,k}} \tag{7}$$

are bounded by 1 in absolute value. As any nonzero eigenvalue of M must also be an eigenvalue of 7, the eigenvalues of M are bounded by 1 in absolute value as well.

For each $\tau \in \Delta$ define Ψ_τ as

$$\Psi_\tau(X) = \sum_{k} A_{\tau,k} X A_{\tau,k}^*$$

for every $X \in \mathbb{C}^{N \times N}$. For $t \geq 1$ we have

$$\begin{aligned} M^{t+1}[N^2 + 2, 1] &= vec\left(\sum_{k} A_{\tau_0,k}^* A_{\tau_0,k}\right)^{\mathsf{T}} \left(\sum_{\tau \neq \tau_0} \sum_{k} A_{\tau,k} \otimes \overline{A_{\tau,k}}\right)^{t-1} vec(\rho) \\ &= \sum_{\tau_1,\ldots,\tau_{t-1} \in \Delta \setminus \{\tau_0\}} \mathrm{Tr}\left(\Psi_{\tau_0} \circ \Psi_{\tau_{t-1}} \circ \cdots \circ \Psi_{\tau_1}(\rho)\right) \\ &= \sum_{\tau_1,\ldots,\tau_{t-1} \in \Delta \setminus \{\tau_0\}} \Pr[R_1 = \tau_1, \ldots, R_{t-1} = \tau_{t-1}, R_t = \tau_0] \\ &= \Pr[R_1 \neq \tau_0, \ldots, R_{t-1} \neq \tau_0, R_t = \tau_0]. \end{aligned}$$

It is obvious that $M[N^2 + 2, 1] = -\beta$, and so the lemma is proved. □

Now we have all of the required pieces to prove Theorem 20.

Proof. Proof of Theorem 20 Following from the fact that the sets $\{\mathcal{E}_x\}$ and $\{\rho_x\}$ have $\mathrm{GapL}_{\mathbb{A}}$-descriptions, we may assume that we have a positive integer polynomial m, a finite collection $\{\alpha_1, \ldots, \alpha_n\}$ of algebraic numbers, and GapL functions a, b, c, and d such that each operation $\mathcal{E}_x$ is described by the $p(|x|) \times p(|x|)$ matrices $\{A_{\tau,k} : \tau \in \Delta, 1 \leq k \leq m(|x|)\}$ given by

$$A_{\tau,k}[i,j] \;=\; \sum_{l=1}^{n} \frac{a(x,\tau,k,i,j,l)}{b(x,\tau,k,i,j,l)}\,\alpha_l$$

for every $x \in \Sigma^*$, $\tau \in \Delta$, $1 \leq k \leq m(|x|)$, and $1 \leq i,j \leq p(|x|)$, and

$$\rho_x[i,j] = \sum_{l=1}^{n} \frac{c(x,i,j,l)}{d(x,i,j,l)}\,\alpha_l$$

for every $x \in \Sigma^*$ and $1 \leq i,j \leq p(|x|)$. Here, we are assuming (without loss of generality) that the same set $\{\alpha_1, \ldots, \alpha_n\}$ is used in the $\mathrm{GapL}_{\mathbb{A}}$-descriptions of both $\{\mathcal{E}_x\}$ and $\{\rho_x\}$. We may also assume without loss of generality that $\{\mathcal{E}_x\}$ and $\{\rho_x\}$ are real by Lemma 21, and therefore that $\{\alpha_1, \ldots, \alpha_n\}$ are real algebraic numbers. (If $\{\mathcal{E}_x\}$ and $\{\rho_x\}$ are not real, then a simple modification of a, b, c, and d allows the real and imaginary parts of $\alpha_1, \ldots, \alpha_n$ to be combined in a manner consistent with Lemma 21.)

Let M_x be the $(p(|x|)^2 + 2) \times (p(|x|)^2 + 2)$ matrix described in Lemma 22 for each input x. Using Theorem 6 it is routine to define GapL functions f and g such that

$$M_x[i,j] \;=\; \sum_{l=1}^{n+1} \frac{f(x,i,j,l)}{g(x,i,j,l)}\,\alpha_l,$$

where we write $\alpha_{n+1} = \beta$ in order to simplify notation. By Lemma 22 we see that the series

$$\sum_{t\geq 0} M_x^t[p(|x|)^2 + 2, 1]$$

must converge, as the sum is $-\beta$ plus a sum over probabilities of mutually exclusive events. Moreover, Lemma 22 implies that

$$\Pr[\exists\, t : R_{x,t} = \tau_0] > \beta$$

if and only if

$$\sum_{t\geq 0} M_x^t[p(|x|)^2 + 2, 1] > 0.$$

As the family $\{M_x\}$ has a $\mathrm{GapL}_{\mathbb{A}}$-description, the theorem now follows from Lemma 14. □

6 Simulating space-bounded quantum computations

In this section, we apply Theorem 20 to the quantum Turing machine model defined in Section 2.3. This will establish Theorem 1, which is restated here for convenience.

Theorem 1. *For any space-constructible function $s(n) \in \Omega(\log n)$, we have*

$$\mathrm{PrQSPACE}_{\mathbb{A}}(s) = \mathrm{PrSPACE}(s).$$

It should be stressed at this point that although the application of Theorem 20 to the quantum Turing machine model defined in Section 2.3 of course requires one to focus on specifics of this model, this application is fairly routine—we claim that essentially any other "reasonable" model of space-bounded quantum computation can be handled using the same general method. The purpose of discussing the specific quantum Turing machine model defined previously is mainly intended to demonstrate this idea.

6.1 Logarithmic space-bounds

We first prove Theorem 1 for the special case where $s(n)$ is logarithmic in n. In Section 6.2 this fact is extended to arbitrary space-constructible space bounds using a standard padding argument.

Theorem 24. $\text{PrQSPACE}_{\mathbb{A}}(\log) = \text{PL}$.

Proof. Let $L \subseteq \Sigma^*$ be any language in $\text{PrQSPACE}_{\mathbb{A}}(\log)$, and let M be a quantum Turing machine running in space $s(n) \in O(\log n)$ that recognizes L and satisfies the properties of Definition 9. The notation of Section 2.3 will be used to describe the various parts of M as needed; for example, the state set of M is Q, the classical transition function of M is δ, and the quantum operations of M are given by $\{\mathcal{E}_q : q \in Q\}$. Each operation $\mathcal{E}_q$ is determined by a collection

$$\left\{ A_{\tau,k}^{(q)} : \tau \in \Delta,\, 1 \leq k \leq m_0 \right\} \tag{8}$$

of $2^{K+1} \times 2^{K+1}$ matrices with algebraic number entries.

We will construct, for each input x, a selective quantum operation $\mathcal{F}_x$ and an initial state ρ_x as follows. Each operation $\mathcal{F}_x$ will have 3 possible outputs, *accept*, *reject*, and *running*, and the selective quantum process $\{R_{x,1}, R_{x,2}, \ldots\}$ induced by $\mathcal{F}_x$ and ρ_x will represent the computation of M on input x in the following sense:

$$\begin{aligned}
\Pr[R_{x,t} = \textit{accept}] &= \Pr[M \text{ accepts } x \text{ after } t \text{ or fewer steps}],\\
\Pr[R_{x,t} = \textit{reject}] &= \Pr[M \text{ rejects } x \text{ after } t \text{ or fewer steps}],\\
\Pr[R_{x,t} = \textit{running}] &= \Pr[M \text{ is still running on input } x \text{ after } t \text{ steps}].
\end{aligned}$$

Thus

$$\Pr[M \text{ accepts } x] = \Pr[\exists t : R_{x,t} = \textit{accept}].$$

The assumptions on M stated above will imply that the selective quantum operation $\mathcal{F}_x$ and the initial state ρ_x will satisfy the assumptions of Theorem 20. Taking $\beta = 1/2$ will therefore allow us to conclude that $L \in \text{PL}$.

The operation $\mathcal{F}_x$ will act on density matrices whose entries correspond to space $s(|x|)$ classical configurations of M. For each input string x, let S_x be a set defined as follows:

$$S_x = Q \times \{0,1\}^K \times \Delta \times (\Gamma \cup \{\#\})^{s(|x|)} \times \{0,1\}^{s(|x|)} \times \{0,\ldots,|x|+1\} \times \{1,\ldots,s(|x|)\}^2.$$

The elements of S_x represent space $s(|x|)$ classical configurations of M in the following way: $c = (q, y, \tau, w, z, h_1, h_2, h_3)$ represents the configuration of M for which the internal state is $q \in Q$, the classical state of the quantum register is $y \in \{0,1\}^K$, the measurement register contains $\tau \in \Delta$, the $s(|x|)$ squares on the work tape that are reachable in a space $s(|x|)$ computation are described by $w \in (\Gamma \cup \{\#\})^{s(|x|)}$, the classical state of the $s(|x|)$ qubits on the quantum tape that are reachable in a space $s(|x|)$ computation are described by $z \in \{0,1\}^{s(|x|)}$, and the locations of the input tape head, the work tape head, and the quantum tape head are given by h_1, h_2, and h_3. At any point in the computation of M on input x, the mixed quantum state of M can be represented by a density matrix $\rho \in \mathbf{D}(\mathbb{C}^{S_x \times S_x})$.

The operation $\mathcal{F}_x$ will describe one step in the evolution of M on input x. Each step consists of a quantum phase, which is determined by the operations $\{\mathcal{E}_q : q \in Q\}$, and a classical phase, which is determined by the classical transition function δ.

We begin with the quantum phase. The operations $\{\mathcal{E}_q : q \in Q\}$ act on a finite portion of M; specifically, on the K qubits in the internal quantum register and the qubit being scanned on the quantum tape. Each operation $\mathcal{E}_q$ is described by the collection of $2^{K+1} \times 2^{K+1}$ matrices as in 8. We need to show how these constant-size matrices translate to matrices indexed by S_x specifying the global evolution of M on input x. This is routine, but somewhat cumbersome with regard to notation. It can be done as follows.

For each $c \in S_x$, let $\mathrm{state}(c) \in Q$ denote the internal state of M in configuration c, and let $\mathrm{qubits}(c) \in \{0,1\}^{K+1}$ denote the length $K+1$ binary string representing the classical state of the qubits in the internal quantum register together with the qubit being scanned on the qubit tape in configuration c. For example, if $c = (q, y, \tau, w, z, h_1, h_2, h_3)$, then $\mathrm{state}(c) = q$ and $\mathrm{qubits}(c) = yz_{h_3}$. For any string $u \in \{0,1\}^{K+1}$ and any symbol $\tau \in \Delta$, let $\mathrm{replace}_{u,\tau}(c)$ denote the element of S_x obtained by replacing the classical state of the K qubits in the quantum register by the first K bits of u, replacing the classical state of the qubit being scanned on the quantum tape by the last bit of u, and replacing the contents of the measurement register by τ. For each $\tau \in \Delta$ and $1 \leq k \leq m_0$ define $B_{\tau,k} \in \mathbb{C}^{S_x \times S_x}$ as follows.

$$B_{\tau,k}[d,c] = \begin{cases} A_{\tau,k}^{(\mathrm{state}(c))}[\mathrm{qubits}(d), \mathrm{qubits}(c)] & \text{if } r_{\mathrm{qubits}(d),\tau}(c) = d \\ 0 & \text{otherwise.} \end{cases}$$

We have

$$\sum_{\tau,k} B_{\tau,k}^* B_{\tau,k} = I,$$

which follows from the fact that

$$\sum_{\tau,k} \left(A_{\tau,k}^{(q)} \right)^* A_{\tau,k}^{(q)} = I$$

for each $q \in Q$.

Now, the matrices

$$\{B_{\tau,k} : \tau \in \Delta,\, 1 \leq k \leq m_0\} \subseteq \mathbb{C}^{S_x \times S_x}$$

represent the quantum part of each step in the following way: if the mixed state of M is $\rho \in \mathbb{C}^{S_x \times S_x}$ and the quantum phase of a computation step is performed, the resulting mixed state will be

$$\sum_{\tau,k} B_{\tau,k} \rho B_{\tau,k}^*.$$

(The collection $\{B_{\tau,k} : \tau \in \Delta, 1 \leq k \leq m_0\}$ is not interpreted as inducing a selective quantum operation in this case—there is no classical output for this operation.)

The second phase of each computation step is the classical phase. We need to do essentially the same thing we did for the quantum phase of each step. Because the classical part of each step is deterministic, this phase is simpler to describe. The three possible outputs *accept*, *reject*, and *running* will be handled as well.

The transition function δ determines, for any fixed input x, a unique successor for each $c \in S_x$. (If the internal state is a halting state, then the successor of a configuration is defined to be itself.)

Write $c \vdash_x d$ whenever d is the successor of c with respect to this function for input x. Define three sets J_{acc}, J_{rej}, $J_{run} \subseteq S_x \times S_x$ as follows:

$$\begin{aligned} J_{acc} &= \{(c,d) \,:\, c \vdash_x d,\, d \text{ is accepting}\} \\ J_{rej} &= \{(c,d) \,:\, c \vdash_x d,\, d \text{ is rejecting}\} \\ J_{run} &= \{(c,d) \,:\, c \vdash_x d,\, (c,d) \notin J_{acc} \cup J_{rej}\}. \end{aligned}$$

Let $E_{c,d}$ denote the matrix with a 1 in the (c,d) entry and 0 in all other entries.

The global evolution for the classical phase can be described in terms of the matrices $\{E_{c,d}\}$ for pairs (c,d) in J_{acc}, J_{rej}, and J_{run} in a fairly straightforward way. It will, however, be more convenient for us to combine these matrices with the matrices $\{B_{\tau,k}\}$ describing the quantum phase directly. For each pair $(c,d) \in S_x \times S_x$, define

$$C_{accept,(c,d,\tau,k)} = \begin{cases} E_{d,c} B_{\tau,k} & \text{if } (c,d) \in J_{acc} \\ 0 & \text{otherwise,} \end{cases}$$

$$C_{reject,(c,d,\tau,k)} = \begin{cases} E_{d,c} B_{\tau,k} & \text{if } (c,d) \in J_{rej} \\ 0 & \text{otherwise,} \end{cases}$$

$$C_{running,(c,d,\tau,k)} = \begin{cases} E_{d,c} B_{\tau,k} & \text{if } (c,d) \in J_{run} \\ 0 & \text{otherwise,} \end{cases}$$

and let $\mathcal{F}_x$ be the selective quantum operation given by the collection

$$\left\{ C_{v,(c,d,\tau,k)} \,:\, v \in \{accept, reject, running\},\, (c,d,\tau,k) \in S_x \times S_x \times \Delta \times \{1,\ldots,m_0\} \right\}.$$

The operation $\mathcal{F}_x$ acts on $S_x \times S_x$ density matrices and corresponds to one step in the evolution of M as described above.

The initial density matrix ρ_x is much simpler to describe; we let $\rho_x = E_{c,c}$ for c the initial configuration of M.

Because s is logarithmic, $|S_x|$ is bounded by $p(|x|)$ for some positive integer polynomial p. Choose such a polynomial p, and also let m be a positive integer polynomial such that

$$m(|x|) \geq m_0\,|\Delta|\,|S_x|^2 + 1$$

for all $x \in \Sigma^*$.

Next, associate with each element of S_x an integer in the range $\{1,\ldots,|S_x|\}$ by ordering the elements of S_x lexicographically within each coordinate, where the finite sets Q, Γ, Δ, etc., are ordered arbitrarily. (It is not crucial to use this specific ordering—any numbering that is computationally simple will do.) Extend each of the matrices $C_{v,(c,d,\tau,k)}$ as well as ρ_x to $p(|x|) \times p(|x|)$ matrices by respecting this numbering and assigning the value 0 to entries with indices outside of the range $\{1,\ldots,|S_x|\}$.

In a similar manner, associate a unique integer in the range $\{1,\ldots,m(|x|)\}$ with each 4-tuples $(c,d,\tau,k) \in S_x \times S_x \times \Delta \times \{1,\ldots,m_0\}$, and relabel the matrices

$$\{C_{accept,(c,d,\tau,k)}\},\ \{C_{reject,(c,d,\tau,k)}\},\ \text{and } \{C_{running,(c,d,\tau,k)}\}$$

as

$$\{C_{accept,l}\},\ \{C_{reject,l}\},\ \text{and } \{C_{running,l}\},$$

for $l = 1, \ldots, m_0 \, |\Delta| \, |S_x|^2$ by respecting this numbering. In order to ensure that we have a selective quantum operation we must add an additional matrix

$$C_{accept,l}[i,j] = \begin{cases} 1 & \text{if } i = j > |S_x| \\ 0 & \text{otherwise} \end{cases}$$

for $l = m_0 \, |\Delta| \, |S_x|^2 + 1$. Finally, let $C_{v,l}$ be the zero matrix for all remaining values of $v \in \{accept, reject, running\}$ and $l \in \{1, \ldots, m(|x|)\}$.

It is, at this point, routine to observe that $\{\mathcal{F}_x\}$ and $\{\rho_x\}$ (after these re-labellings and extensions) have $\mathrm{GapL}_{\mathbb{A}}$-descriptions. The finite collection of algebraic numbers are those numbers appearing as entries of the matrices $\{A^{(q)}_{\tau,k}\}$, and the coefficients of these numbers are given by ratios of GapL functions. (In this case, in fact, these coefficients only take values 0 and 1, and can be given by an FL function.) This completes the proof. □

6.2 General space-bounds

Finally, we use Theorem 24 to prove Theorem 1. This proof is based on a standard padding argument. Readers interested in a more general discussion of this method are referred to Section 6.4 of [WW86].

Proof. Proof of Theorem 1 Let $L \in \mathrm{PrQSPACE}_{\mathbb{A}}(s)$ and let M_1 be a quantum Turing machine running in space $O(s)$ that witnesses this fact.

Define a new language $L_{\log}$ as follows:

$$L_{\log} = \left\{ x\,0\,1^{2^{s(|x|)}} \;\middle|\; x \in L \right\}.$$

(If the symbols 0 and 1 are not elements of the alphabet over which L is a language, they can be replaced by different symbols or $L_{\log}$ can be defined over a larger alphabet—it does not make any difference to the proof.) We have that $L_{\log} \in \mathrm{PrQSPACE}_{\mathbb{A}}(\log)$. To see this, define a new quantum Turing machine M_2 as follows. The machine M_2 first processes the input (classically), looking for the suffix $0\,1^{2^{s(|x|)}}$. If this suffix is present, it stores the starting location of this suffix so that it may simulate M_1 on the prefix x, accepting or rejecting accordingly. If the suffix $0\,1^{2^{s(|x|)}}$ is not present, M_2 rejects. The presence of the suffix can easily be determined in logarithmic space, using the fact that $s(|x|)$ is space-constructible along with the fact that the machine M_2 can be programmed to reject when it would otherwise try to use more than some appropriate logarithmic amount of space.

As $L_{\log} \in \mathrm{PrQSPACE}_{\mathbb{A}}(\log)$, it follows from Theorem 24 that $L_{\log} \in \mathrm{PL}$.

Finally, we need to argue that $L_{\log} \in \mathrm{PL}$ implies $L \in \mathrm{PrSPACE}(s)$. This is essentially the reverse of the argument above that $L_{\log} \in \mathrm{PrQSPACE}_{\mathbb{A}}(\log)$. Let M_3 be an ordinary probabilistic Turing machine recognizing $L_{\log}$ in logarithmic space. We want to define a new probabilistic Turing machine M_4 that, on input x, simulates M_3 on the input $x\,0\,1^{2^{s(|x|)}}$. Using the space-constructibility of s this is again routine. The machine M_4 marks off $s(|x|) + 1$ squares on its work tape, which allows it to count up to $2^{s(|x|)} + 2$. This counter can be used in the simulation of M_3 on input $x\,0\,1^{2^{s(|x|)}}$ to store the location of the input tape head of M_3 when it is in the region to the right of the string x; otherwise M_4 uses its own input tape, which contains just x. The amount of space needed for this simulation is $O(s)$, and thus $L \in \mathrm{PrSPACE}(s)$. □

7 Conclusion

It has been demonstrated in this paper that quantum computation and classical probabilistic computation are essentially equivalent in the unbounded error, space bounded setting, which implies that quantum computations can be simulated deterministically with a quadratic increase in space. This result requires the quantum machines being simulated to have algebraic number transition amplitudes, but places no restrictions on the sorts of measurements or other non-unitary operations permitted during their computations.

Perhaps the most interesting direction for future work on space-bounded quantum computation regards bounded-error quantum computations (that halt absolutely, i.e., halt with certainty after some finite number of steps). We conclude with two open questions along these lines.

1. Can a bounded-error quantum Turing machine running in space s be simulated deterministically in space $O(s^{2-\varepsilon})$ for some constant $\varepsilon > 0$?
2. Give an example of a natural combinatorial problem that can be solved in logarithmic space by a bounded-error quantum Turing machine that is not known to be solvable by a bounded error classical probabilistic Turing machine in logarithmic space.

Acknowledgements

I would like to thank Richard Cleve for the description of selective quantum processes that appears in Section 2.2, Eric Allender for a helpful discussion on GapL functions, and Marcus Schaefer and Pradyut Shah for their suggestions regarding the proof of Theorem 11. This research was supported by Canada's NSERC.

References

[AAM03] E. Allender, V. Arvind, and M. Mahajan. Arithmetic complexity, Kleene closure, and formal power series. *Theory of Computing Systems*, 36:303–328, 2003.

[ADH97] L. Adleman, J. DeMarrais, and M. Huang. Quantum computability. *SIAM Journal on Computing*, 26(5):1524–1540, 1997.

[AKN98] D. Aharonov, A. Kitaev, and N. Nisan. Quantum circuits with mixed states. In *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, pages 20–30, 1998.

[AO96] E. Allender and M. Ogihara. Relationships among PL, #L, and the determinant. *RAIRO – Theoretical Informatics and Applications*, 30:1–21, 1996.

[BCP83] A. Borodin, S. Cook, and N. Pippenger. Parallel computation for well-endowed rings and space-bounded probabilistic machines. *Information and Control*, 58:113–136, 1983.

[BDE$^+$98] D. Bruss, D. DiVincenzo, A. Ekert, C. Fuchs, C. Macchiavello, and J. Smolin. Optimal universal and state-dependent quantum cloning. *Physical Review A*, 57(4):2368–2378, 1998.

[FFK94] S. Fenner, L. Fortnow, and S. Kurtz. Gap-definable counting classes. *Journal of Computer and System Sciences*, 48:116–148, 1994.

[FGHP99] S. Fenner, F. Green, S. Homer, and R. Pruim. Determining acceptance possibility for a quantum computation is hard for the polynomial hierarchy. *Proceedings of the Royal Society, London A*, 455:3953–3966, 1999.

[For97] L. Fortnow. Counting complexity. In L. Hemaspaandra and A. Selman, editors, *Complexity Theory Retrospective II*, pages 81–107. Springer, 1997.

[FR99] L. Fortnow and J. Rogers. Complexity limitations on quantum computation. *Journal of Computer and System Sciences*, 59(2):240–252, 1999.

[Gil77] J. Gill. Computational complexity of probabilistic Turing machines. *SIAM Journal on Computing*, 6(4):675–695, 1977.

[Isa94] I. M. Isaacs. *Algebra: a Graduate Course*. Brooks/Cole, 1994.

[Jun85] H. Jung. On probabilistic time and space. In *Proceedings of the 12th International Colloquium on Automata, Languages and Programming*, volume 194 of *Lecture Notes in Computer Science*, pages 310–317. Springer-Verlag, 1985.

[KSV02] A. Kitaev, A. Shen, and M. Vyalyi. *Classical and Quantum Computation*, volume 47 of *Graduate Studies in Mathematics*. American Mathematical Society, 2002.

[Mah61] K. Mahler. *Lectures on Diophantine Approximations*, volume 1. Cushing Malloy, 1961.

[NC97] M. Nielsen and C. Caves. Reversible quantum operations and their application to teleportation. *Physical Review A*, 55(4):2547–2556, 1997.

[NC00] M. A. Nielsen and I. L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 2000.

[Sak96] M. Saks. Randomization and derandomization in space-bounded computation. In *Proceedings of the 11th Annual IEEE Conference on Computational Complexity*, pages 128–149, 1996.

[Sav70] W. Savitch. Relationships between nondeterministic and deterministic tape complexities. *Journal of Computer and System Sciences*, 4:177–192, 1970.

[TD00] B. Terhal and D. DiVincenzo. On the problem of equilibration and the computation of correlation functions on a quantum computer. *Physical Review A*, 61: article 022301, 2000.

[Val79] L. Valiant. The complexity of computing the permanent. *Theoretical Computer Science*, 8:189–201, 1979.

[Wat99] J. Watrous. Space-bounded quantum complexity. *Journal of Computer and System Sciences*, 59(2):281–326, 1999.

[WW86] K. Wagner and G. Wechsung. *Computational Complexity*. Mathematics and Its Applications. D. Reidel Publishing Company, 1986.